```
 *   p :: payload byte
 *
 */

extern void fake();                    /*  fake predefined func to init lt table */
extern void null();                    /*  null func for unused function entries */

typedef struct {                       /*  lt table structure */
  char length;                         /* block length or length of length field  */
  void (* function)();                 /* predefined function for lt 0..5 */
  char id offset;                      /* offset to ident field for lt 6..15 */
} Lt entry;

Lt entry lt table[16] = {                          /* lt table declaration */
  {2, fake,  0}, {2, fake,  0}, {4, fake,  0}, {5, fake,  0},      /*  0..3    */
  {6, fake,  0}, {0, fake,  0}, {1, null,  3}, {1, null,  4},      /*  4..7    */
  {2, null,  4}, {2, null,  5}, {4, null,  6}, {4, null,  8},      /*  8..11   */
  {6, null,  8}, {6, null, 10}, {0, null,  0}, {0, null,  0}       /* 12..15  */
};

typedef struct {                       /* id table structure     */
  char length;                         /* length of ident field   */
  char table;                          /* which table to use in lookup */
} Id_entry;

Id_entry id_table[8] = {               /* id table declaration    */
  {1, 0}, {1, 1}, {2, 0}, {3, 0}, {2, 2}, {4, 2}, {8, 2}, {0, 0}
};

extern void (* lookup())();            /* id lookup function    */


/*  EXECUTABLE CODE STARTS HERE  */

int decode compact header(pkt)         /* call standard decoder and return length */
    unsigned char *pkt;                /* pointer to packet          */
{
  unsigned char *p = pkt;              /* working pointer to packet  */
  int lt;                              /* length type                */
  int length;                          /* block length               */
  Id entry *pid;                       /* id table pointer           */
  void (* f)();                        /* standard function returned from lookup  */

  lt = *p >> 4;                        /* get length type from bits 4..7 of key */
  if (lt < 6)                          /* process predefined formats: lt < 6   */
    {
      length = lt table[lt].length;    /*get block length from lt table*/
      (* lt table[lt].function)(p);    /* call predefined function    */
      return length;
    }

  /* extract enough bytes to cover length field and shift out unused bits  */
```

```
  length = ((int)*(p+2)<<24) | ((int)*(p+3)<<16) | ((int)*(p+4)<<8) | *(p+5);
  length >>= 32-(lt table[lt].length * 8);

  pid = &id table[*p & 0x7];          /*get id table ptr using bits 0..2 of key */
  p += lt table[lt].id offset;        /* move pointer to start of id field   */
  f = lookup(p,pid->length,pid->table);   /* lookup function */
  p += pid->length                    /* move pointer to start of payload */
  (* f)(p,length-(int)(p-pkt));      /*call function with payload ptr,length*/
  return length;
}
```

B.2 ASN.1 Header Decoder

The following program decodes a packet with an ASN.1 header. The ASN.1 header is a compatible subset of the standard ASN.1 EXTERNAL type. As stated in section 6., not all the flexibility of the standard EXTERNAL type is needed to meet the objectives. Thus this decoder has been specialized to support a level of function roughly comparable to that of the compact header decoder.

Decoding an ASN.1 header is performed by parsing a sequence of tokens. Block length is defined by one sequence, standard identification by a second. The header has short or extended forms. The short form is used for blocks of 128 bytes or less and is processed by extracting block length from a single byte. The extended form is processed by constructing block length from multiple bytes.

The identifier is left as a string of bytes used to compute a hash table lookup of a decoding function. The hash table lookup is performed by the procedure lookup() which takes identifier address and length and returns a pointer to a corresponding function (f).

B.2.1 Cautionary Notes

Context dependent headers are not decoded by the code below. They are decoded by standard decode procedures at a time they are expected.

Block length is assumed to fit within one 32 bit word. Extending the program and/or the C language to support larger word sizes, thus larger block lengths, is possible and likely to happen as 64 bit processor architectures emerge.

If an unknown identifier is encountered, the lookup function will return a pointer to an appropriate default function that ignores the payload and displays an informative message.

B.2.2 Program Text

The program has two parts---the first part contains variable and procedure declarations, the second part (at the end) contains the dozen or so statements actually executed. Throughout the code descriptive notes (comments that are

not executed) are placed between comment delimiters (/*...*/).

```
/* ASN.1 header has one of three forms:
 *
 * Each character in the strings below represents a byte; bytes between
 * square brackets are optional; payload bytes are not counted
 *
 * 2 byte (minimum) header for context dependent messages:
 *
 *   "tl[...l]"
 *
 * 7 byte (minimum) header for short blocks:
 *
 *   "tltli[...i]tl"
 *
 * Extended header for longer blocks:
 *
 *   "tl[...l]tli[...i]tl[...l]"
 *
 * Key:
 *
 *   t :: tag byte
 *   l :: length byte
 *   i :: id byte
 *
 */

/*   EXECUTABLE CODE STARTS HERE  */

decode_asn1_header(pkt)       /* call standard decoder and return length   */
    unsigned char *pkt;             /* pointer to packet */
{
 extern void (* lookup())();        /* id lookup function */
 unsigned char    *p = pkt;         /* working pointer to packet */
 unsigned int length;               /* block length */
 void (* f)();                     /* standard function returned from lookup*/
 unsigned int t;                    /* temp register */

 length = *(p+1);                   /* get 1st length byte */
 if (length < 127)                  /* look for short form */
  {
    t = *(p+3);                     /* get id field length */
    f = lookup(p+4,t);              /* lookup function */
    (* f)(p+t+6,length-(p+t+6-pkt)); /* call function  */
    return length + 2;              /* return total length */
  }
 t = length - 128;                  /* calc length of length field */

 /*  extract enough bytes to cover length field and shift out unused bits   */

 length = ((int)*(p+2)<<24) | ((int)*(p+3)<<16) | ((int)*(p+4)<<8) | *(p+5);
 length >>= 32-(t * 8);
```

```
    p += t + 3;                     /* move pointer to start of id field */
    f = lookup(p+1,*p);             /* lookup function  */
    p += *p + t + 3;                /* move pointer to start of payload  */
    (* f)(p,length-(int)(p-pkt));   /* call func (payload ptr and size) */
    return length + t + 2;          /* return total length */
}
```

APPENDIX C

HEADER/DESCRIPTOR TASK FORCE MEMBERS

Will Stackhouse, Chairman	Jet Propulsion Laboratory
David H. Staelin, Vice Chairman	Massachusetts Institute of Technology
Walter Bender	Massachusetts Institute of Technology
Rita Brennan	Apple Computer, Inc.
Wayne E. Bretl	Zenith Electronics
David C. Carver*	Digital Equipment Corporation
Gary Demos	Demografx
Ephraim Feig	IBM
Branko Gerovac*	Digital Equipment Corporation
Jeffrey W. Johnston	Eastman Kodak Company
Michael Liebhold	Apple Computer Inc.
Lee McKnight	Massachusetts Institute of Technology
Arun Netravali	AT&T Bell Laboratories
Bruce Sidran	Bellcore
D. Scott Silver	Tektronix, Inc.
Richard J. Solomon	Massachusetts Institute of Technology
David L. Tennenhouse	Massachusetts Institute of Technology
David L. Trzcinski	PictureTel
Ken C. Yang	Ampex Corporation

* Associate members making significant written contributions.